

1. What is CICD?

Continuous Integration and continuous Delivery (CI/CD) is a set of software practices and techniques that enable the frequent release of small batches of code changes, with extensive visibility and traceability. It typically involves the creation of a largely automated pipeline that orchestrates the build, test and deployment of software across staged environments, ultimately leading to deployment in production.

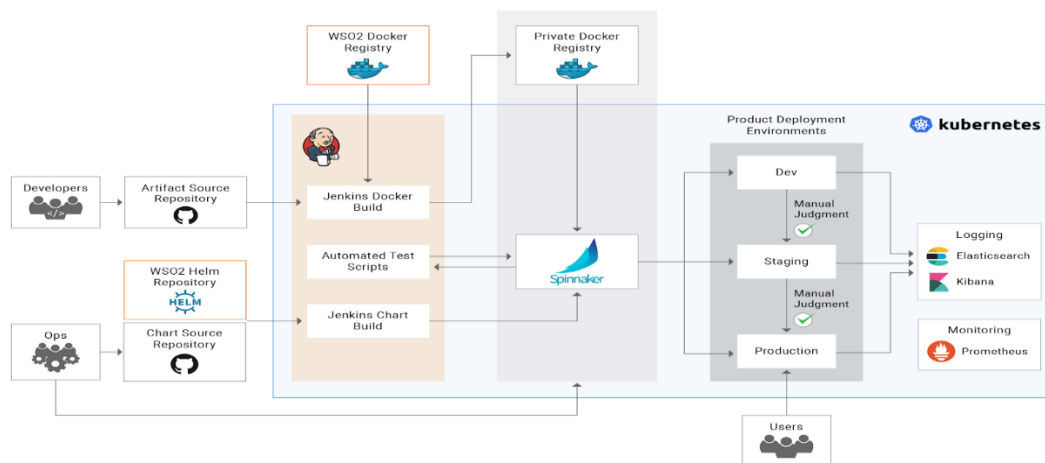
2. What are all the CICD tools you are using?

- ✓ GitHub
- ✓ Jenkins
- ✓ Soap
- ✓ Junit
- ✓ Nexus
- ✓ Kibana
- ✓ App Dynamics
- ✓ PCF (Pivotal Cloud Foundry)
- ✓ Gradle

3. Difference between Continuous integration, Continuous Delivery and Continuous Deployment?



4. Explain how CICD works internally?



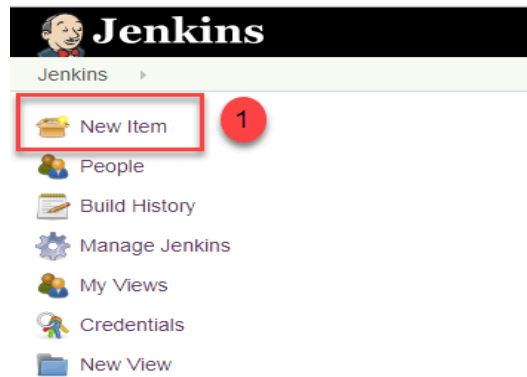
5. How to create the Jenkins jobs?

✓ Login to Jenkins



The image shows the Jenkins login page. At the top is the Jenkins logo and name. Below it is a navigation bar. The main content area contains a login form with two input fields: 'User:' and 'Password:'. A red box labeled '1' highlights the 'User:' field. Below the password field is a checkbox labeled 'Remember me on this computer'. A red box labeled '2' highlights the 'log in' button.

✓ Click on new Item




✓ Enter the Item Details


- Enter the name of the item you want to create. We shall use the "Hello world" for this demo.
- Select Freestyle project
- Click Okay


Enter an item name


Enter an item name
Hello World 1


* Required field


 **Freestyle project**
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any tool used for something other than software build. 2

 **Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipeline and/or organizing complex activities that do not easily fit in free-style job type.

 **Multi-configuration project**
Suitable for projects that need a large number of different configurations, such as testing on multiple builds, etc.

 **Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, separate namespace, so you can have multiple things of the same name as long as they are in different folders.

 **GitHub Organization**
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

 **Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository. 3

OK

✓ Enter Project Details

General Source Code Management Build Triggers Build Environment Build Post-build Actions

Description
Hello world java test program

[Plain text] [Preview](#)

☐ Discard old builds

☐ GitHub project

☐ This project is parameterized

☐ Throttle builds

☐ Disable this project

☐ Execute concurrent builds if necessary

[Advanced...](#)

✓ Enter repository URL (Git Hub)

If your GitHub repository is private, Jenkins will first validate your login credentials with GitHub and only then pull the source code from your GitHub repository.

General **Source Code Management** Build Triggers Build Environment Build Post-build Actions

Source Code Management

☐ None

☒ Git

Repositories

Repository URL https://github.com/kirru/firstJava.git

Credentials - none - [Add](#)

[Advanced...](#)

[Add Repository](#)

Branches to build

Branch Specifier (blank for 'any') */master

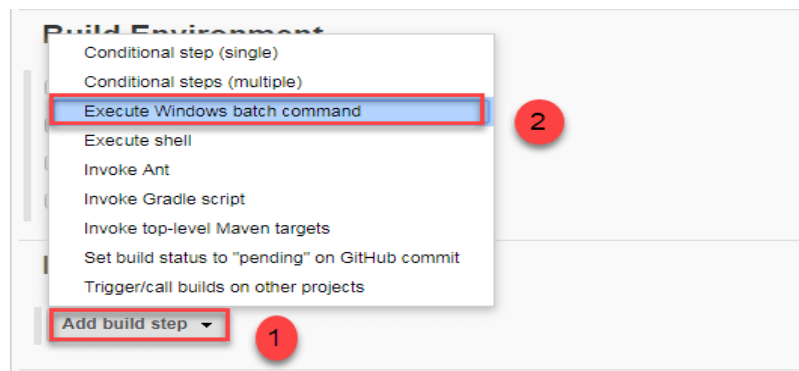
[Add Branch](#)

✓ Tweak the settings

Now that you have provided all the details, it's time to build the code. Tweak the settings under the build section to build the code at the time you want. You can even schedule the build to happen periodically, at set times.

Under build,

- Click on "Add build step"
- Click on "Execute Windows batch command" and add the commands you want to execute during the build process.



- ✓ Here, I have added the java commands to compile the java code.
- ✓ I have added the following windows commands:

```
javac HelloWorld.java  
java HelloWorld
```



- ✓ Build Source code

Back to Dashboard

Status

Changes

Workspace

Build Now

Delete Project

Configure

Rename

Project Hello World

Hello world java test program

Workspace

Recent Changes

Permalinks

✓ Check the console output for build status.

Jenkins > Hello World > #1

Back to Project

Status

Changes

Console Output

View as plain text

Edit Build Information

Delete Build

Next Build

Console Output

Started by user The_Guru99

Building in workspace C:\Program Files (x86)\Jenkins\workspace\Hello World

Cloning the remote Git repository

Cloning repository <https://github.com/kriru/firstJava.git>

> git.exe init C:\Program Files (x86)\Jenkins\workspace\Hello World # timeout=10

Fetching upstream changes from <https://github.com/kriru/firstJava.git>

> git.exe --version # timeout=10

> git.exe fetch --tags --progress <https://github.com/kriru/firstJava.git> +refs

> git.exe config remote.origin.url <https://github.com/kriru/firstJava.git> # t:

> git.exe config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/'

> git.exe config remote.origin.url <https://github.com/kriru/firstJava.git> # t:

Fetching upstream changes from <https://github.com/kriru/firstJava.git>

> git.exe fetch --tags --progress <https://github.com/kriru/firstJava.git> +refs

> git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10

> git.exe rev-parse "refs/remotes/origin/origin/master^{commit}" # timeout=10

> git.exe rev-parse "origin/master^{commit}" # timeout=10

C:\Program Files (x86)\Jenkins\workspace\Hello World>javac HelloWorld.java

C:\Program Files (x86)\Jenkins\workspace\Hello World>java HelloWorld

Hello World

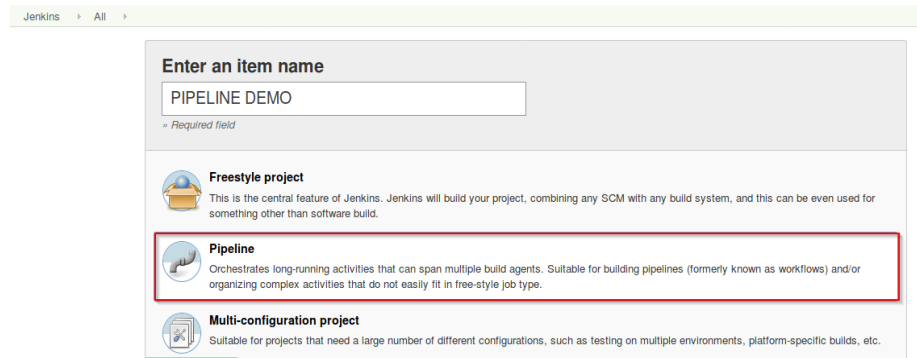
Finished: SUCCESS

6. How to create the pipeline job using Jenkins?

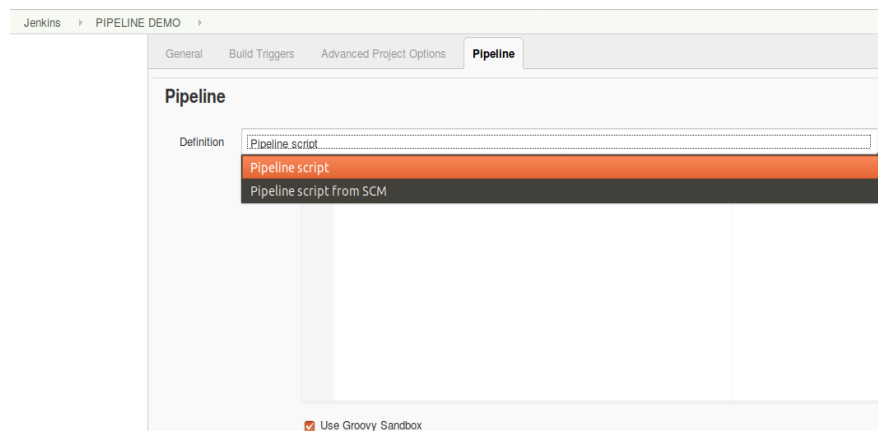
✓ Click on new Item



- ✓ Enter a name for your pipeline and select 'pipeline' project. Click on 'ok' to proceed.



- ✓ Scroll down to the pipeline and choose if you want a declarative pipeline or a scripted one.



- ✓ If you want a scripted pipeline then choose 'pipeline script' and start typing your code.

Jenkins > PIPELINE DEMO >

General Build Triggers Advanced Project Options **Pipeline**

Pipeline

Definition **Pipeline script**

Script 1 // TYPE YOUR CODE HERE try sample Pipeline...

☒ Use Groovy Sandbox

- ✓ If you want a declarative pipeline then select 'pipeline script from SCM' and choose your SCM. In my case I'm going to use Git. Enter your repository URL.

Jenkins > PIPELINE DEMO >

General Build Triggers Advanced Project Options **Pipeline**

Pipeline

Definition **Pipeline script from SCM**

SCM **Git**

Repositories

Repository URL **https://github.com/Zulaikha12/git-test.git**

Please enter Git repository.

Credentials **- none -** Add

Advanced...

Add Repository

Add Branch

- ✓ Within the script path is the name of the Jenkinsfile that is going to be accessed from your SCM to run. Finally click on 'apply' and 'save'. You have successfully created your first Jenkins pipeline.

Jenkins > PIPELINE DEMO >

General Build Triggers Advanced Project Options **Pipeline**

Repository browser **(Auto)**

Additional Behaviours **Add**

Script Path **Jenkinsfile**

Lightweight checkout ☒

[Pipeline Syntax](#)

Save Apply

7. Which kind of Jenkins pipe line you are using for building the release ear?

We are using Cirrus pipe line along with trident compliance for below phases,

- ✓ Check out the code from Git repository
- ✓ Execute the Junits
- ✓ Do the sonar validations code repository
- ✓ Skip the build if Junits/Sonar validations got failed.
- ✓ Do the added jars compliance with Nexus IQ server. If any critical compliance trigger the mail for Job requestor.
- ✓ Validate the Integration test cases
- ✓ Validate the Acceptance test cases
- ✓ For validating the acceptance
- ✓ Mock the corresponding service response with **wire mock** and deploy the web and service mock applications in PCF Zones.
- ✓ After completion of the validations on all api's with wire mock remove the deployed mock apps from PCF.
- ✓ Run the Pen test to identify the security vulranabilities in code.
- ✓ Prepare the release ear and move it to Nexus.
- ✓ In the final step Trident collect all the above evidences and move those as zip folder to nexus and GitHub.

8. Which kind of Jenkins pipe line you are using for deploying the release ear?

We are using Blue-Green deployment for deployment to maintain the zero downtime.

At any time, only one of the environments is live, with the live environment serving all production traffic. For this example, Blue is currently live and Green is idle.

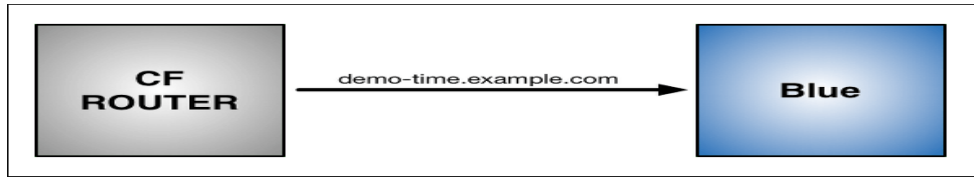
As you prepare a new version of your software, deployment and the final stage of testing takes place in the environment that is not live: in this example, Green. Once you have deployed and fully tested the software in Green, you switch the router so all incoming requests now go to Green instead of Blue. Green is now live, and Blue is idle.

Blue-Green deployment follow below steps,

- ✓ Push an App
- ✓ Update App and Push
- ✓ Map Original Route to Green
- ✓ Unmap Route to Blue
- ✓ Remove Temporary Route to Green

Step1: Push an App:

Blue is now running on Cloud Foundry. The CF Router sends all traffic for demo-time.example.com traffic to Blue.

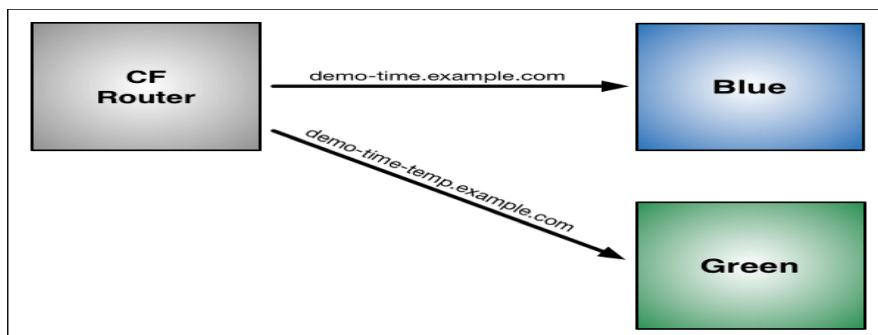


Step2: Update App and Push:

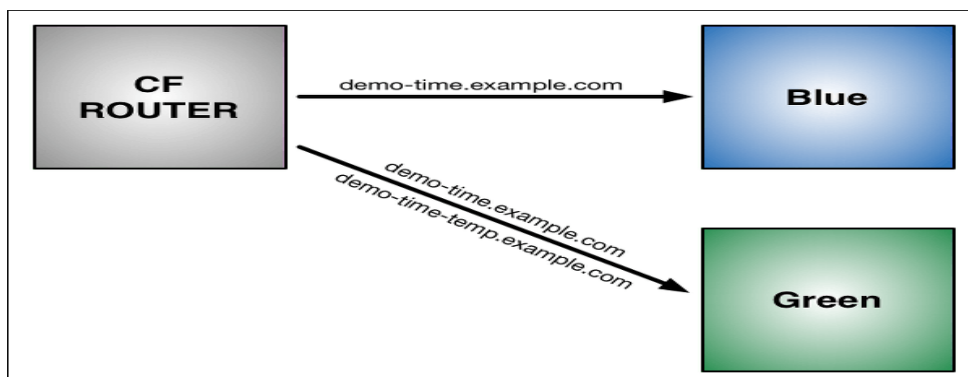
Now make a change to the app. First, replace the word "Blue" on the web page with "Green," then rebuild the source file for the app. Run cf push again, but use the name "Green" for the app and provide a different subdomain to create a temporary route,

Two instances of our app are now running on Cloud Foundry: the original Blue and the updated Green.

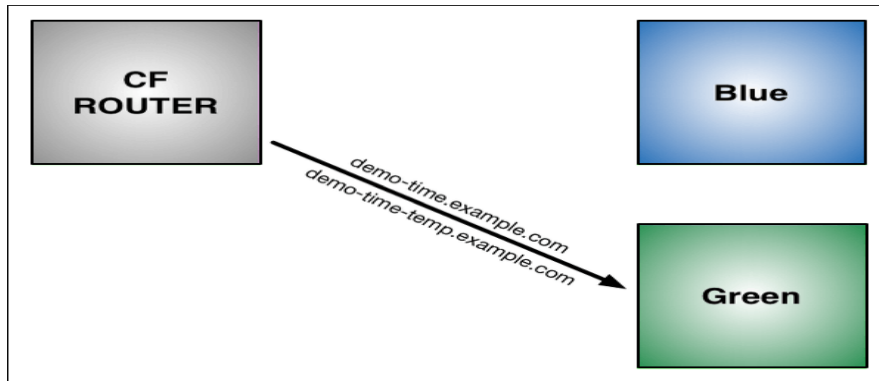
The CF Router continues sending all traffic for demo-time.example.com to **Blue**. The router now also sends any traffic for demo-time-temp.example.com to **Green**.



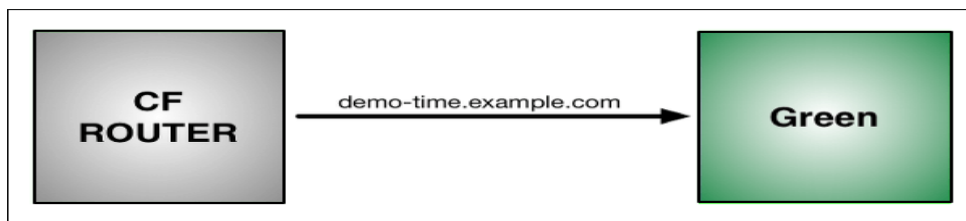
Step3: Map Original Route to Green



Step4: Unmap Route to Blue

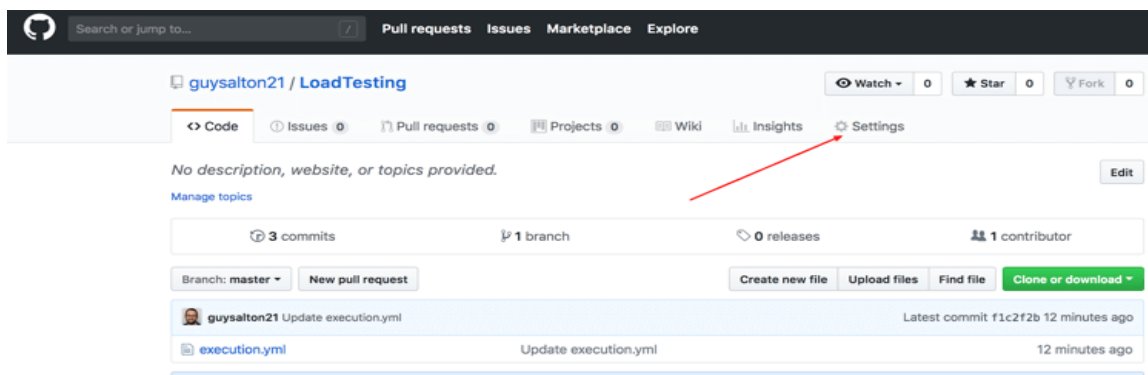


Step5: Remove Temporary Route to Green

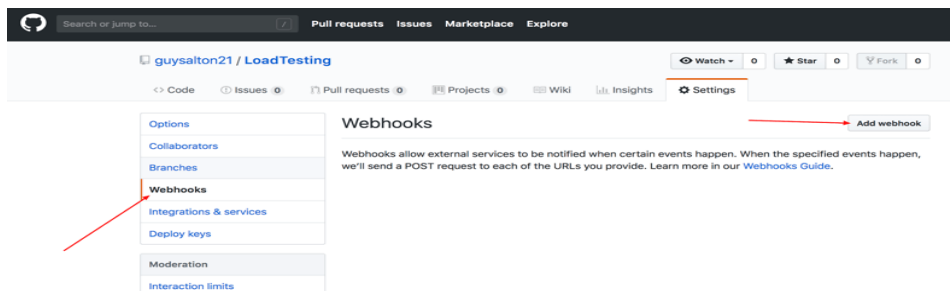


9. Explain the steps to do the once click deployment on code commit from github?

Step1: Go to your GitHub repository and click on 'Settings'



Step2: Click on Webhooks and then click on 'Add webhook'.



Step3: In the 'Payload URL' field, paste your Jenkins environment URL. At the end of this URL add /github-webhook/. In the 'Content type' select: 'application/json' and leave the 'Secret' field empty.

The screenshot shows the GitHub 'Add webhook' configuration page. On the left is a sidebar with navigation links: Options, Collaborators, Branches, Webhooks (highlighted), Integrations & services, Deploy keys, Moderation, and Interaction limits. The main content area is titled 'Webhooks / Add webhook' and includes a description: 'We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in our developer documentation.' Below this, the 'Payload URL' field contains 'http://jenkins-demo.blazemeter.com:8080/github-webhook/'. The 'Content type' dropdown is set to 'application/json'. The 'Secret' field is empty. At the top right, there are buttons for 'Watch', 'Star', and 'Fork', each with a count of 0. Below the repository name, there are tabs for 'Code', 'Issues', 'Pull requests', 'Projects', 'Wiki', 'Insights', and 'Settings' (which is active).

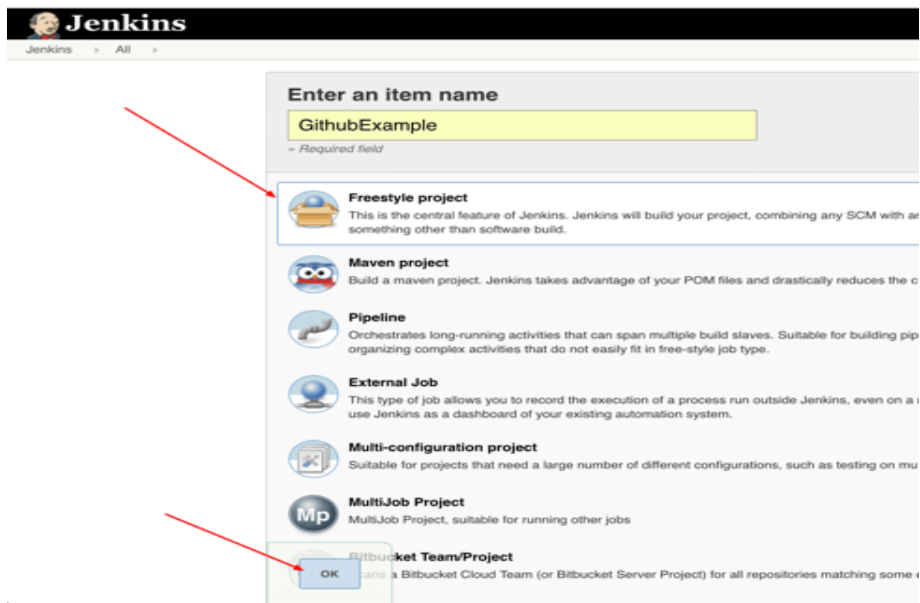
Step4: In the page 'Which events would you like to trigger this webhook?' choose 'Let me select individual events.' Then, check 'Pull Requests' and 'Pushes'. At the end of this option, make sure that the 'Active' option is checked and click on 'Add webhook'.

The screenshot shows the event selection page for a GitHub webhook. It features a list of event categories with checkboxes and descriptions. Red arrows point to the 'Pull requests' checkbox, the 'Pushes' checkbox, the 'Active' checkbox, and the 'Add webhook' button. The events listed are: Visibility changes (Repository changes from private to public.), Pull request reviews (Pull request review submitted, edited, or dismissed.), Pull requests (checked) (Pull request opened, closed, reopened, edited, assigned, unassigned, review requested, review request removed, labeled, unlabeled, or synchronized.), Pull request review comments (Pull request diff comment created, edited, or deleted.), Releases (Release published in a repository.), Repositories (Repository created, deleted, archived, unarchived, publicized, or privatized.), Repository imports (Repository import succeeded, failed, or cancelled.), Repository vulnerability alerts (Vulnerability alert created, resolved, or dismissed on a repository.), Statuses (Commit status updated from the API.), Team adds (Team added or modified on a repository.), Watches (User stars a repository.), and Active (checked) (We will deliver event details when this hook is triggered.). A green 'Add webhook' button is at the bottom.

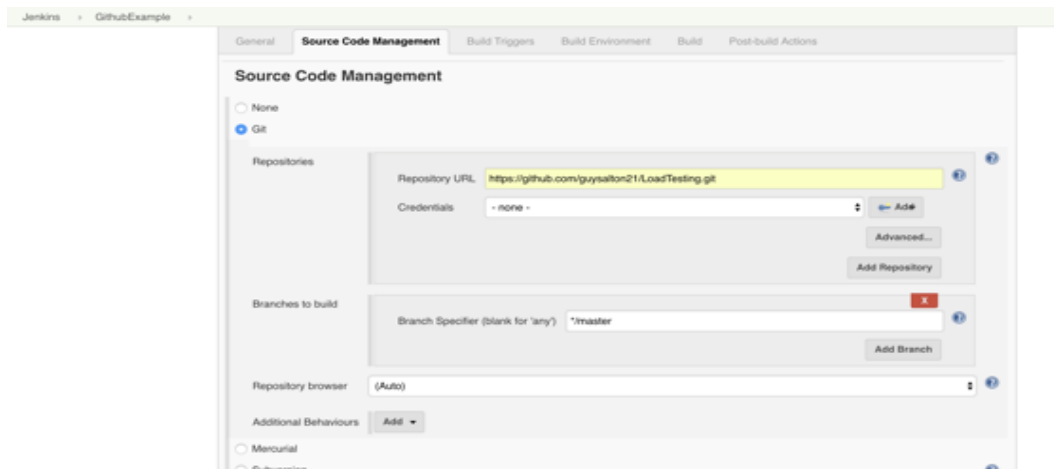
Step6: In Jenkins, click on 'New Item' to create a new project.



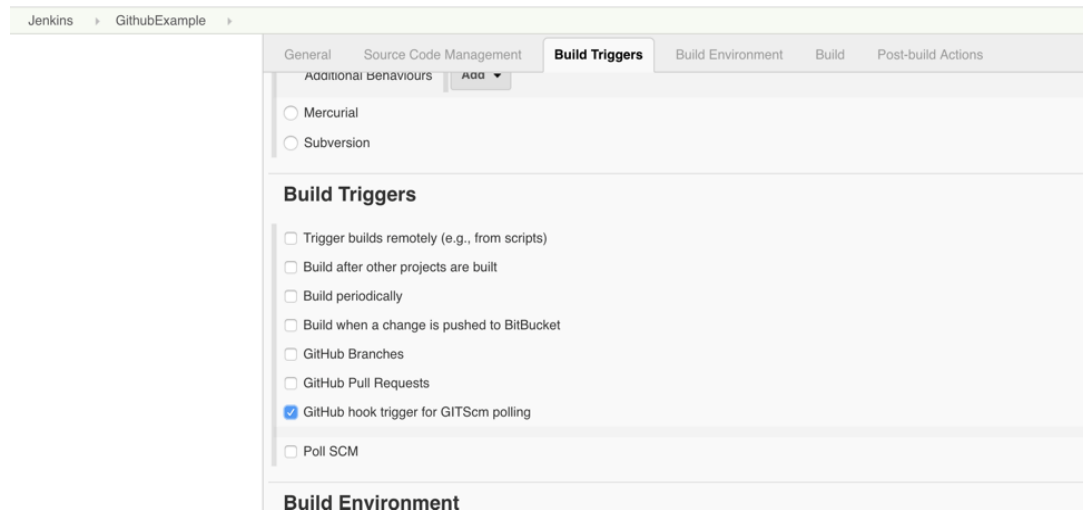
Step 7: Give your project a name, then choose 'Freestyle project' and finally, click on 'OK'.



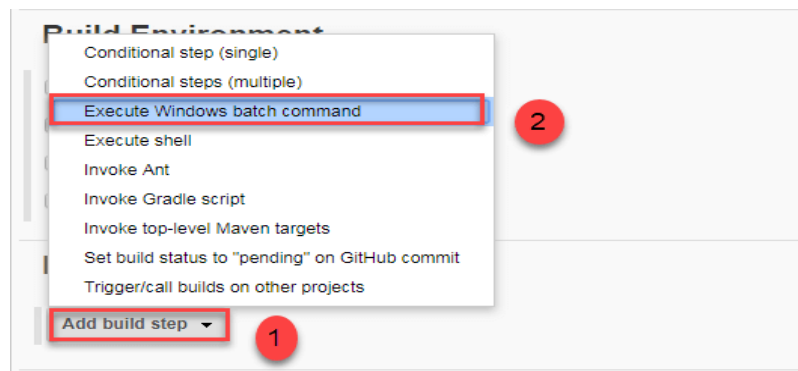
Step 8: Click on the 'Source Code Management' tab select git and give the github url in it



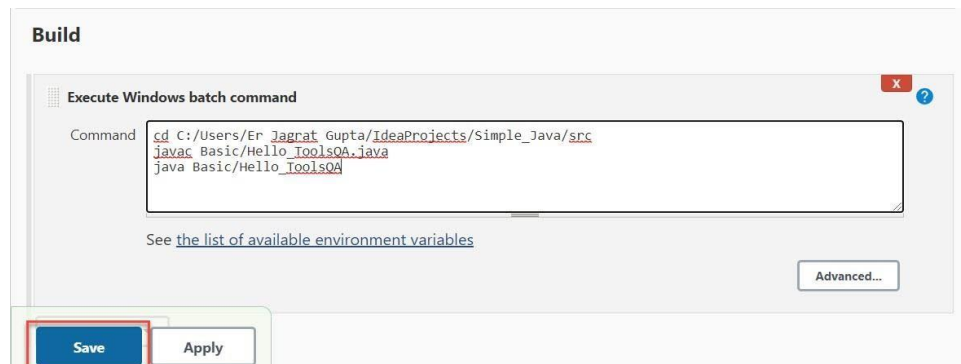
Step 9: Click on the 'Build Triggers' tab and then on the 'GitHub hook trigger for GITScm polling'. Or, choose the trigger of your choice.



Step 10: Click on the 'Build' tab, then click on 'Add build step' and choose 'Execute windows batch command'.

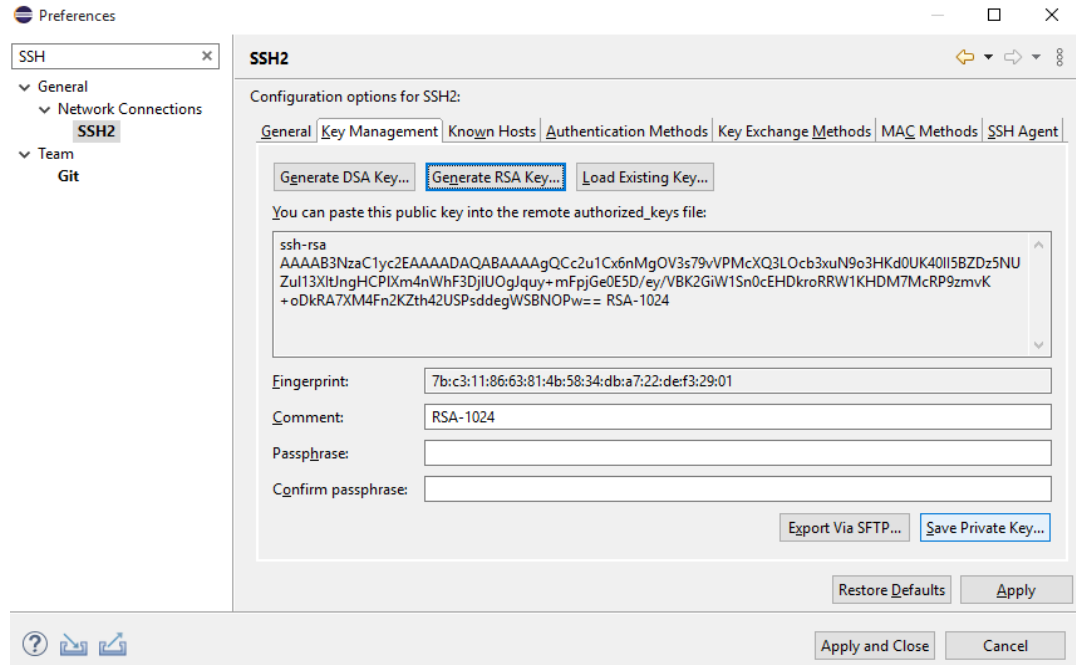


Step 11: In the build give the commands to be executed on your source code



10. How to setup the path to check in the code from local to github?

- ✓ In the Eclipse preference tab select the SSH2. Click on Key Management and then click on Generate RSA Key.

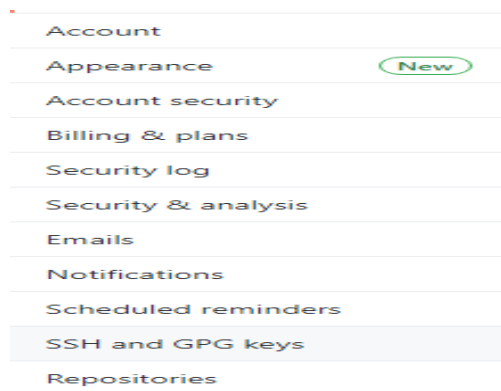


- ✓ Save the Above private key to your system

This PC > Local Disk (C:) > Users > home > .ssh

Name	Date modified	Type	Size
id_rsa	18-Nov-20 06:46 P...	File	1 KB
id_rsa.pub	18-Nov-20 06:46 P...	Microsoft Publish...	1 KB
known_hosts	12-Mar-21 05:54 PM	File	2 KB

- ✓ Go to the github settings and select the SSH and GPG keys




- ✓ Click on new SSH key and paste the key generated from eclipse here then click on save.

SSH keys

[New SSH key](#)

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.



SSH
SHA256:ukjIpZ5kd9upiuaeT+iNT1xUJUH+I5tihtpgTZ7P1dw
Added on Nov 18, 2020
Last used within the last week — Read/write

Delete

11. Explain the limitations on CICD?

Security Vulnerabilities: CI/CD pipelines have a lot of advantages for deploying software builds through an automated process. Unfortunately, a lot of cyber attackers will target CI/CD pipelines because there are usually some security vulnerabilities that are not addressed. A lot of sensitive information can be vulnerable within a CI/CD pipeline, which is why hackers will try to gain access to the pipeline through extreme measures.

Performance Issues: The goal of any CI/CD pipeline will likely be to deliver software and code updates as fast as possible through an automated process. The problem is that a lot of performance issues can make their way into the software if things are not done properly.

Some developers like the idea of implementing an automated testing system to check for possible performance issues. For example, if a batch of code is not performing at an efficient rate, a warning might be pushed back to the developers for additional evaluation. This can help create a defense system against the potential release of a poorly performing software build to customers.

Communication: You will most likely be working with quite a few other people if you are working within a CI/CD pipeline. You might even be broken up into different teams with varying responsibilities. Sometimes the biggest issue within a CI/CD is human communication. For example, if something fails during a software deployment, communication is going to be imperative to solving the issue in a timely manner.

It is plausible to assume that a scenario develops where an automated build test outputs an error and then fails to communicate that information to the appropriate developer, there could be some serious consequences. That is only one reason why communication is so important within this industry.

Version Control: A traditional CI/CD pipeline typically requires a lot of components, processes, and resources to be used. When you finally have your CI/CD pipeline working properly, all of your processes are likely functioning on a stable version. If even a single process gets updated unexpectedly, then your entire CI/CD pipeline deployment process could break.